DEVELOPMENT AND IMPLEMENTATION OF A FAST FACTORIZED BACK-PROJECTION CODE TO SPEED UP IMAGE RECONSTRUCTION FOR SYNTHETIC APERTURE RADAR

Senior Honors Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Distinction in the College of Engineering of The Ohio State University

By

Danish U. Siddiqui

Honors Thesis Committee:

Dr. Lee Potter, Advisor

Dr. Rajiv Ramnath

Abstract

Back-projection is a widely used imaging technique used in medical imaging, as well as in synthetic aperture radar (SAR). Conventional back-projection is an extremely time consuming process, with the computational time being proportional to O(N^3), if the reconstructed image is of size $N \times N$. Fast Factorized Back-Projection (FFBP) is an alternate process that significantly reduces the computational time, which is proportional to O($N^2 \log(N)$) in this case. This is achieved by dividing the reconstruction into multiple steps, which leads to a decrease in total computational time.

Acknowledgements

The author would like to thank his ECE 683 group, which includes Taylor Williams and Laura Tufts, whose help and support were instrumental for the successful completion of this thesis project. Thanks also goes to Professor Potter for his mentoring during the project, and his help that ensured a timely completion of the thesis project and its report.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1: Motivation:

In most imaging techniques, time is of essence. The time required to compute traditional Back-Projection is large, being on the order of N^3, where the image is N-by-N pixels. Fast Factorized Back-Projection is a more sustainable imaging technique, as the time that it takes for reconstruction —proportion to N^2 log N -- is much more practical compared to traditional Back-Projection. The goal of this project is to write Matlab codes for Back-Projection and Fast Factorized Back-Projection for Synthetic Aperture Radar geometry, and compare the computational time and image quality of the reconstructed images for both techniques. Along with this, the aim is to also implement Back-Projection for Bi-Static geometry for synthetic data on Matlab, as well as real data.

1.2: Background:

In all imaging techniques, two of the biggest goals are good quality of the reconstructed image, and a time efficient imaging process. In the case of traditional Back-Projection it is found that the reconstructed image has extremely good quality, however the process is extremely time consuming. On the other hand, it is found that Fast Factorized Back-Projection has a tradeoff for these two goals. This tradeoff is such that the quality of the reconstructed image goes down as the time efficiency of the process increases. In this paper, image reconstruction will be analyzed for two different geometries, the first one being Bi-Static, and the second one being Mono-Static.

In the Bi-Static case, the geometry is such that the transmitters and receivers are at different locations. These transmitters and receivers are placed around an object, and the objective is to reconstruct this object. Such a geometry was simulated in Matlab to do

Back-Projection on simulated data, and an experimental test-bed was constructed for a similar geometry to do Back-Projection for real collected data.

In the Mono-Static case, the geometry was simulated to replicate the geometry that is found in Synthetic Aperture Radar (SAR). In SAR, a single transmit/receive element is translated to yield an effective increase in width of the realized aperture; data are recorded at many positions along the synthesized aperture. In this thesis the flight path will be taken to be linear. This geometry was synthesized in Matlab, and to one side of the array of transmitters and receivers would be objects that will be attempted to reconstruct. Using data from Matlab simulator, Back-Projection and Fast Factorized Back-Projection are implemented and compared.

1.3: Impacts:

As mentioned before, traditional Back-Projection is a process that is extremely long computationally. This leads to an issue of sustainability, as if the process takes such a long time, there is the possibility that by the time the output is obtained, it may no longer be useful. Fast Factorized Back-Projection solves this sustainability issue, as it significantly decreases the computational time for the image reconstruction.

1.4: Standards Used:

One of the standards used in this project was the USB 2.0 standard for data communication between the data acquisition system and the PC. This standard affected the rate at which data can be transferred between the two devices, and thus provided an upper bound for the sampling rate beyond which not all data collected can be stored. Thus, this also provided an upper bound to the frequency range of the transmitted waves.

Chapter 2: Hardware

The project included the need of some hardware, including speakers, microphones and a data acquisition system. While choosing these devices, several features were considered. The first one was the cost of the device. The objective was to spend as less as possible for speakers and microphones, while ensuring that the operating frequency range was suitable for the project. Along with this, these speakers and microphones had to be connectable to the data acquisition system, which had to be connected to the computer via USB. Lastly, the data acquisition system would have to have the capability to connect with multiple input and output devices at a time, while having a high enough sampling rate so that a larger frequency range is available.

2.1: Speakers and Microphones:

For the hardware, speakers were used as transmitters, and microphones were used as receivers. The speakers that were used were Neodymium Tweeter's, and the receivers were Electret Microphones (see figure 2). These 'tweeters' or speakers were chosen after the testing of several types of speakers. Unlike the other speakers that were tested, these speakers had a good audio response. They did not have any unexpected noises coming out of them, and the testing determined that within the 2.5 to 30 kHz frequency range, the output of these speakers were fairly consistent. As the microphones worked fairly well and consistently while recording within this frequency range, the transmitted wave that was chosen lay within this frequency range. The results of the testing of the speaker and microphone pair can be seen in Figure 1 below. This testing was done using two methods, one of which is the PN code and the other one being a linear chirp. As can be

seen, the results of both tests were fairly consistent, and therefore these speaker and microphone pair were used in the project.

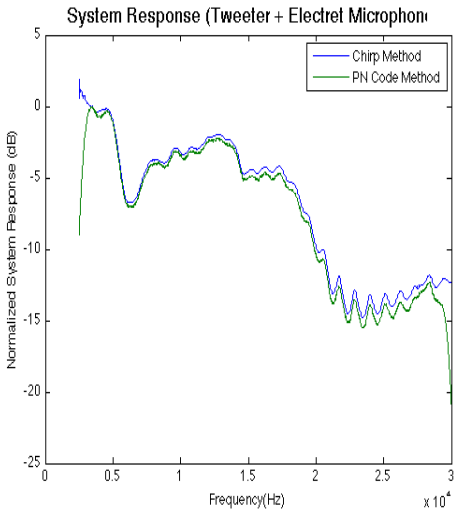

Figure 1: Frequency Response of Tweeter and Electret Microphone Pair

2.2: Data Acquisition System:

The data acquisition system that was used is the Tascam US-1641 (see figure 2). This device, which is an Analog to Digital and Digital to Analog Converter was selected due to its features. Firstly, this device has a sampling rate of 96,000 samples per second, which gave the freedom for the transmit waveform to have a frequency up to 48,000 Hz, which is adequate for this project. Next, as multiple transmitters and receivers have to be used at the same time in the testing phase of the project, the capability of this device to be able to support up to four transmitters and eight receivers at a time is extremely advantageous. Last, as nothing more then a USB connection is needed to transfer data between the device and the PC, this is another extremely useful feature which led to the final choice of the Tascam US-1641 as the data acquisition system used in the project.

Figure 2: Hardware used to transmit and record audio signals

2.3: Imaging Test Stand:

For the Bi-Static case, an imaging test stand was built to collect real data to implement Back-Projection on (see figure 3). This imaging test stand consisted of 4 speakers and 16 microphones, placed in approximately a 1-meter wide circle. The speakers were in 90 degree spacing's, and 4 microphones were placed between each speaker pair. As the Data Acquisition System can only support 8 microphones at a time, first 8 microphones were connected, and an echo was recorded, and then the other 8 microphones were connected, and an echo was again recorded using the same transmit waveform. In this way, recordings were made using all 16 microphones.

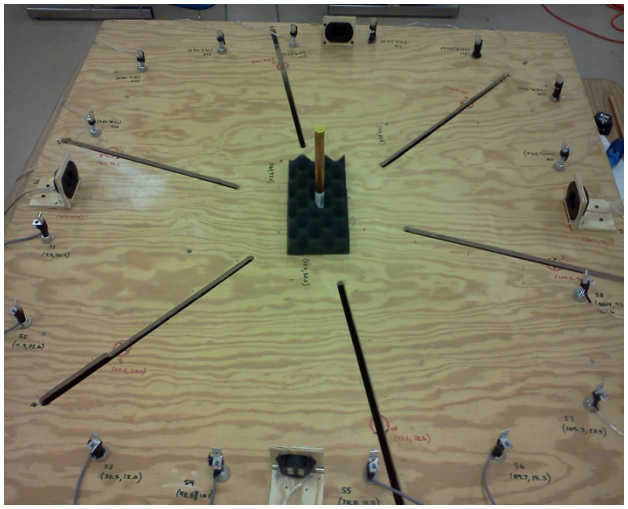

Figure 3: Imaging Test Stand for Bi-Static Imaging

Chapter 3: Back-Projection:

For successful implementation of Back-Projection, there are several steps that need to be undertaken. Firstly, a good transmit waveform is required, which was chosen to be a PN Code. This PN code spanned the 2.5 to 30 kHz frequency range, and consisted of ones and negative ones. The biggest advantage of using such a code as a transmit waveform is that its autocorrelation yields an approximate impulse response.

To do Back-Projection, this PN Code is transmitted, and its echoes from the scene of the image are recorded. The transmission is done for each transmitter in the scene, and recordings are made with each receiver. The received signal is then filtered. This matched filter's impulse response is with the time reversal of the transmitted PN Code; thus, the output is an approximate impulse response of the scene. Next, the background impulse response is subtracted from the output of the matched filter. The background image is the recording done without the presence of any objects, which would contain just the receivers, transmitters, and reflections from the empty test stand. The waveform that is obtained after the background subtraction is the range profile that will be used for the Back-Projection. This process can be seen in Figure 4 below.

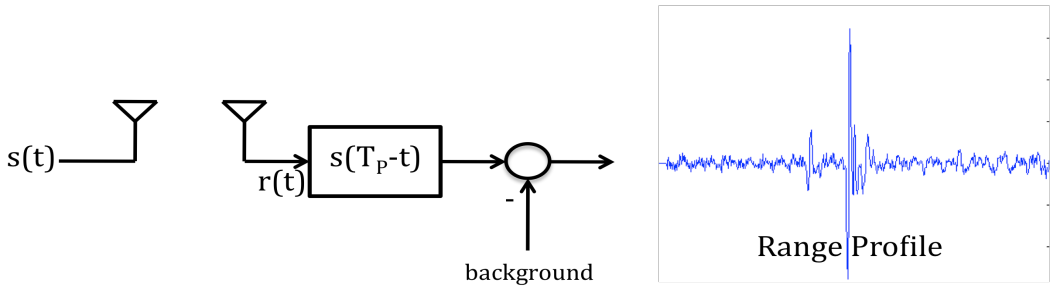

Figure 4: Process to get Range Profile

In the Matlab implementation of Back-Projection, the first step is to interpolate the range profile to fit an image of $N \times N$ size. This is done using the Matlab 'interp1' command,

and is done for each Transmitter and Receiver pair. Next, by using the known positions of the transmitters and receivers, the collected data is Back-Projected to an image of size $N \times N$. This is done using the knowledge that for the recorded data for each receiver, the initial recordings represent the waveforms that have the shortest distance from the transmitter to the receiver, and each subsequent waveform travelled a longer distance. This step is repeated for each transmitter and receiver geometry, and the output represents the reconstructed image. The Back-Projection is identical for both the simulated data as well as the real data; with the only difference being that in the case of the simulated data there will be no noise present, and thus no background subtraction will be required.

Chapter 4: Synthetic Aperture Radar (SAR)/ Mono-Static Transmitter and Receiver Geometry:

"Synthetic Aperture Radar (SAR) is one of the most sophisticated engineering inventions of the 20th century[5]. SAR-radar can either be mounted on an airplane or satellite to take high-resolution images of terrain. SAR has been used since 1950, but due to the lack of computer power and advanced digital signal processing algorithms, the SAR system could not be used in an efficient way [1] at that time.

Pulse compression with a high-bandwidth waveform, such as the matched filtering described Section 3, can be used to obtain a high resolution in distance. But, for high azimuth resolution SAR, coherent integration is required. An example: Azimuth resolution (Δa) for an antenna.

In order to image an area one km (R) away, utilizing a one-meter (D) antenna, with a wavelength of 50 cm (λ), the resolution will be 500 m [2]:

$$\Delta a = \frac{R \times \lambda}{D}$$

Now if the antenna instead were one km long the resolution would become 50 cm. But building a one km long antenna is an impossible task. However if the radar instead were to be moved along a one km synthetic straight line where it sends and receives echoes along the whole synthetic line one would have a one km synthetic radar.

When storing the echo, the exact position from where it originated is not known; only the range and the lobe width are known. The position, in the mono-static case, is specified by a single pulse only as lying on a sphere centered at the transmitter location. Many pulses are recorded and processed to obtain a high-resolution image, as shown in Figure 5 [3]. For the bistatic case, the loci of possible reflection locations become an ellipse with foci

at the transmitter and receiver locations.

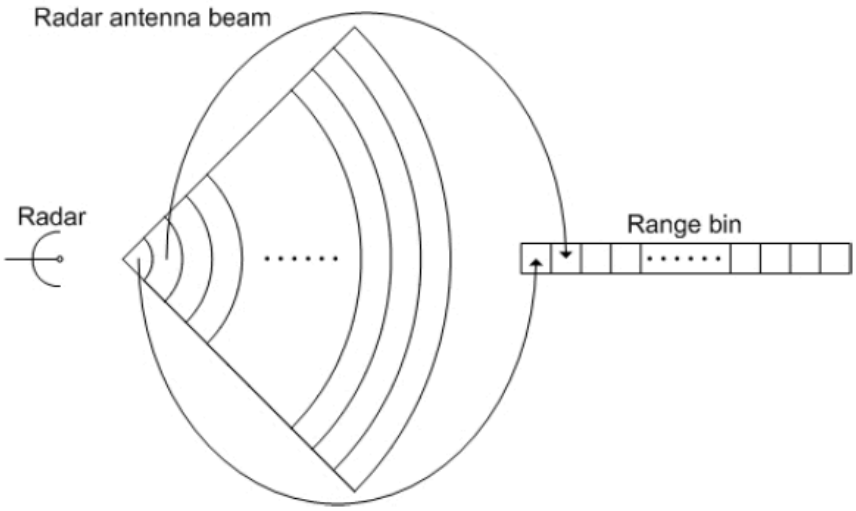

Figure 5: Construction of Range Bins

The range bins are then put together into echo matrices, Figure 6.

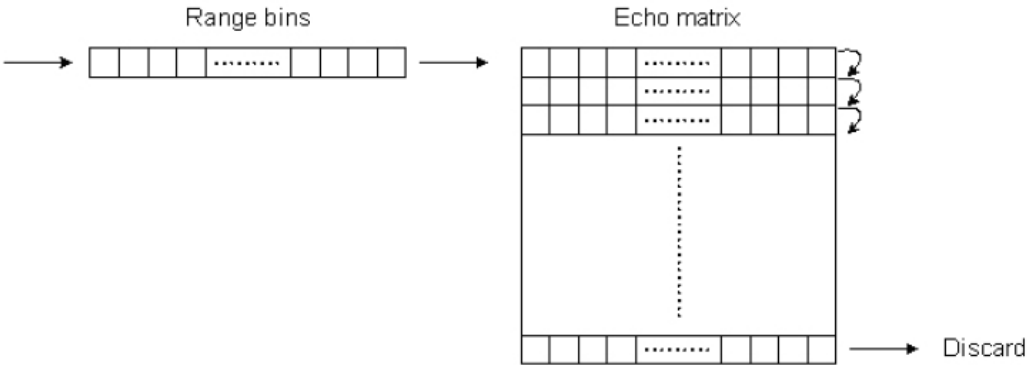

Figure 6: Construction of Echo Matrices

For this method to work the assumption is that the airplane does not move in-between sending and receiving the echoes, this is called the start-stop-approximation. This assumption can be made because the radar waves/echoes move at the speed of light, which is considerably faster than the airplane.

An example:

If an airplane travels at 252 km/h (70 m/s) and is taking images of an area 50 km away

with an angle of 45 degrees in front of the plane, it will take approximately 0.33 ms for the pulse to travel there and back again. Under these 0.33 ms the airplane will travel approximately 2.33cm that means that the airplane/radar will have moved approximately 1.5 cm closer to the target area. This is comparably small to the wavelength that is between three and fifteen meters [2].

SAR has three different operating modes, stripmap, spotlight and scan, Figure 7 [2].

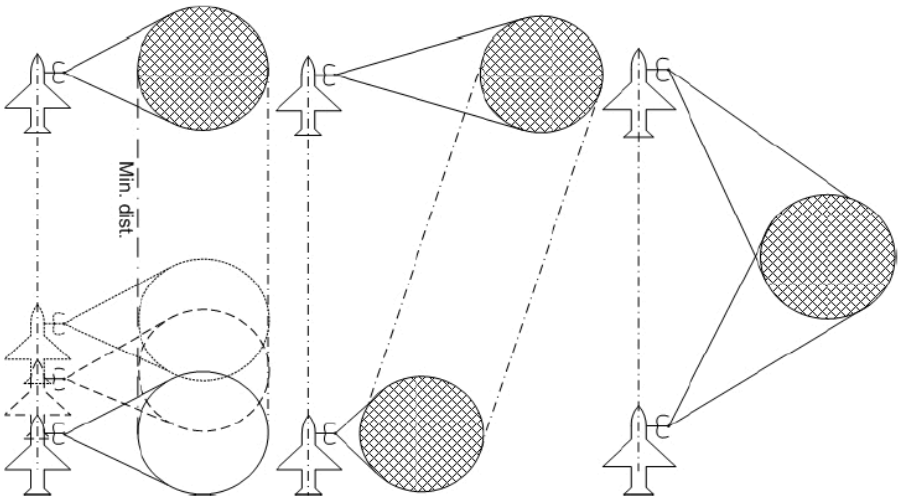

Figure 7: SAR Operating Modes: Stripmap, Spotlight and Scan

Only stripmap will be described in this report. The angular resolution can be interpreted as arising from the different Doppler frequencies experienced at each pulse because of the movement of the radar. With stripmap the resolution depends on the physical lobe width in the manner that the area that is reproduced must be within all of the aperture positions. This means that the synthetic aperture can become longer if the antenna has a larger physical lobe width, which also gives a higher resolution.

The azimuth resolution of the physical aperture decreases according to the distance, but the synthetic apertures resolution can be made independent of distance because of the property of SAR. This means a resolution below one meter can be achieved by airplanes

and satellites from a large distance. On large distances, other limiting factors of the resolution come into play, like the precision of the antenna position, output power of the transmitter, the sensibility of the receiver, disturbances in the lobe expansion and dynamic distance. But at the end, the angular resolution is limited by wavelength, aperture angle and bandwidth as shown below [4].

$$\Delta a_{sar} = \frac{\lambda_c}{2(V_2 - V_1)} \times \frac{c}{2B}$$

The Aperture angle $V_2 - V_1$ is the angle over which the antenna is moved as seen from the image ground, λ_c is the wave length corresponding to the center frequency, $\lambda_c = \frac{c}{f_c}, f_c = \frac{(f_{\max} - f_{\min})}{2}$, c is the speed of light, and B is the bandwidth [4].” [5]

Chapter 5: Fast Factorized Back-Projection (FFBP):

FFBP is an imaging technique that is applicable for mono-static SAR. In this imaging technique, the computational time is much faster compared to traditional Back-Projection. This is done by dividing the collected range bins into multiple range bins, and then interpolating each range bin separately to construct images for each of them. At the end, all the sub-images are added together to get the reconstructed image. This division is known as creating sub-apertures.

To get back the reconstructed image, the division and subsequent addition have to be done correctly. This is explained below for the two and three sub-aperture cases. The decrease in processing time is directly proportional to the number of sub-apertures that are created, however, the quality of the reconstructed image decreases with increase in number of sub-apertures. In this chapter, we adopt the FFBP approach as described in [4] and [5].

5.1: Two Sub-Apertures:

The following parameters and variables are used in subsequent explanations:

Matrix C- Matrix of Collected Data having L columns and r rows

Matrix C1 and C2- Divided Matrices used for Faster Reconstruction having L/2 columns and r rows

Matrix IM1 and IM2- Reconstructed Images for each sub-aperture, size N/2 x N/2

Matrix IM- Final Reconstructed Image, size $N \times N$

L- Length of Collected Data

r- Number of Receivers = Number of Transmitters

$N \times N$- Size of Reconstructed Image

The first step is to divide Matrix C, into Matrices C1 and C2. This is done by putting half the columns into C1, and half the columns into C2. C1 is constructed by placing all the odd numbered columns of C into C1. C2 is constructed by placing all the even numbered columns of C into C2. This is shown in Figure 8.

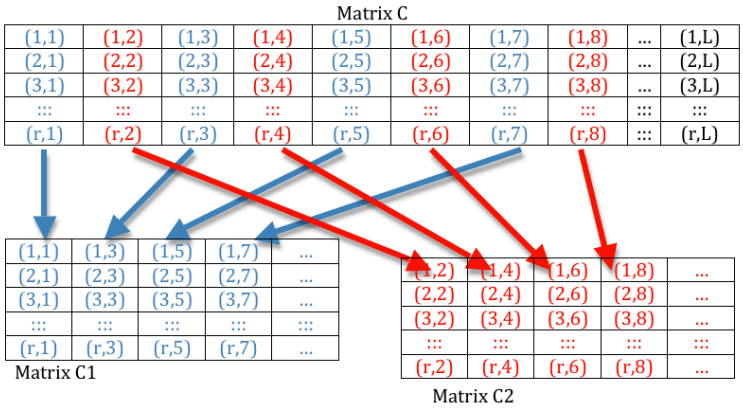

Figure 8: Creating two Sub-Apertures

The next step is to Back-Project and construct images for C1 and C2. We get an image IM1 for the reconstruction of C1, and an image IM2 for the reconstruction of C2. Both of these images are of size N/2 x N/2. Once IM1 and IM2 have been created, the last part is to create IM, which is a $N \times N$ matrix containing the whole scene. IM is created by adding IM1 and IM2, and this is done as shown in Figure 9.

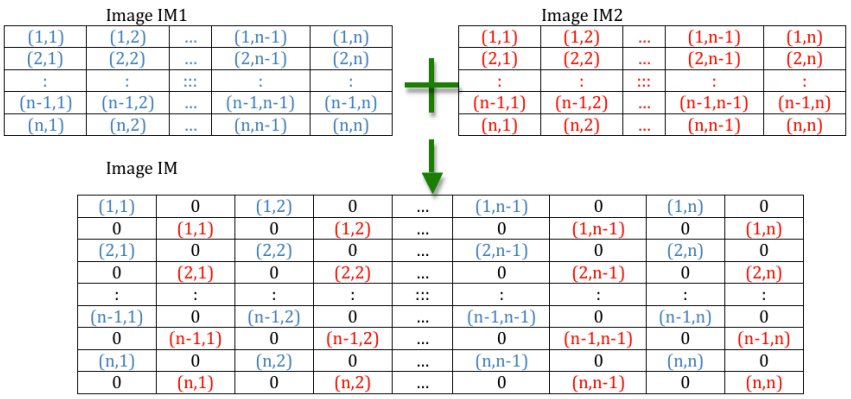

Figure 9: Constructing Final Image for two Sub-Apertures

5.2: Three Sub-Apertures:

Parameters and Variables used in explanation:

Matrix C- Matrix of Collected Data having L columns and r rows

Matrix C1, C2 and C3- Divided Matrices used for Faster Reconstruction having L/3 columns and r rows

Matrix IM1, IM2 and IM3- Reconstructed Images for each sub-aperture, size $N/3 \times N/3$

Matrix IM- Final Reconstructed Image, size $N \times N$

L- Length of Collected Data

r- Number of Receivers = Number of Transmitters

$N \times N$- Size of Reconstructed Image

The first step is to divide Matrix C, into Matrices C1, C2 and C3. This is done by placing the 1st, 4th and every subsequent 3rd column of C into C1, placing every 2nd, 5th and every subsequent 3rd column into C2, and placing every 3rd, 6th and every subsequent 3rd column into C3. This is shown in Figure 10.

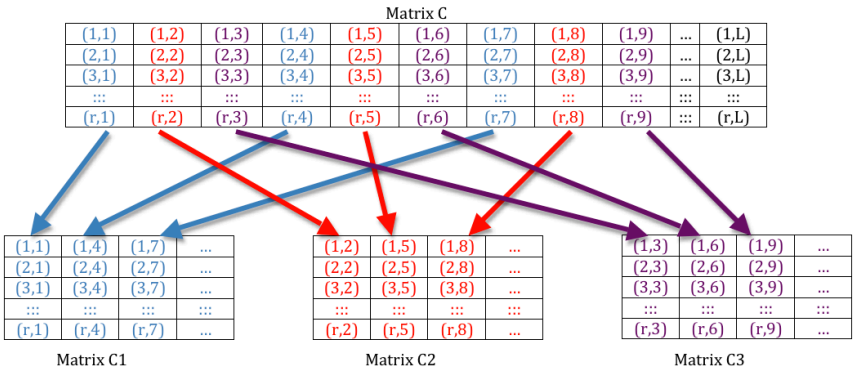

Figure 10: Creating three Sub-Apertures

The next step is to Back-Project and construct images for C1, C2 and C3. We get an image IM1 for the reconstruction of C1, an image IM2 for the reconstruction of C2, and

an image IM3 for the reconstruction of C3. All three of these images are of size $N/3 \times N/3$. Once IM1, IM2 and IM3 have been created, the last part is to create IM, which is a $N \times N$ matrix containing the whole scene. IM is created by adding IM1, IM2 and IM3, and this is done as shown in Figure 11.

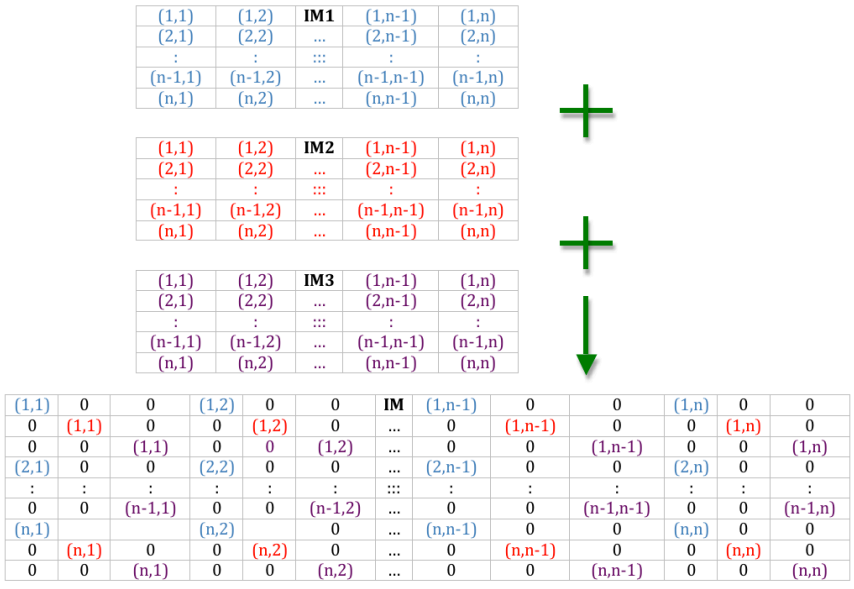

Figure 11: Creating Final Image for three Sub-Apertures

5.3: Generalized Division:

Similar to the processing in the two and three division case, Fast Factorized Back-Projection can be implemented for any number of divisions by following the same process. However, the more sub-apertures we create, the more zero's come into the final reconstructed image, and thus the more the image quality diminishes.

Chapter 6: Testing and Results:

This chapter presents the results of the testing part of the project. Testing was done for the Bi-Static Back-Projection, as well as for the Synthetic Aperture Radar. For both parts, the results obtained are presented, and the reasons for the type of results obtained have been discussed. For the Bi-Static case, testing was done using both synthetic and real data, whereas in the SAR testing were done for only synthetic data. For SAR, results have been presented for traditional Back-Projection, as well as for Fast Factorized Back-Projection for two through eight sub-aperture cases.

6.1: Bi-Static Back-Projection:

Bi-Static Back-Projection was done for both simulated and real data. The simulated data were collected for the geometry shown in the figure below:

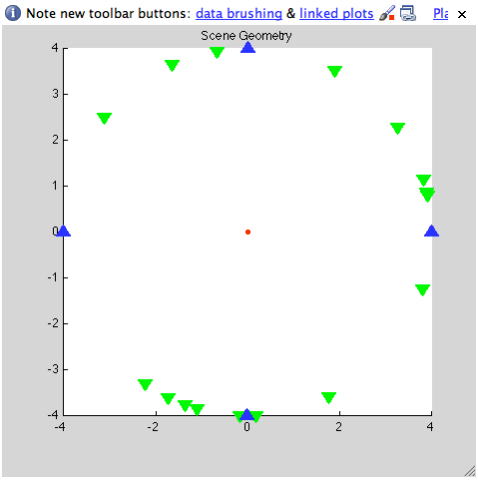

Figure 12: Simulated Test Bed for Bi-Static Imaging with one Reflector

Real data were collected for the image stand that is shown in Figure 2. Back-Projection was done for both these geometries, and an example of the Back-Projection done for the simulated data is shown in Figure 13 below.

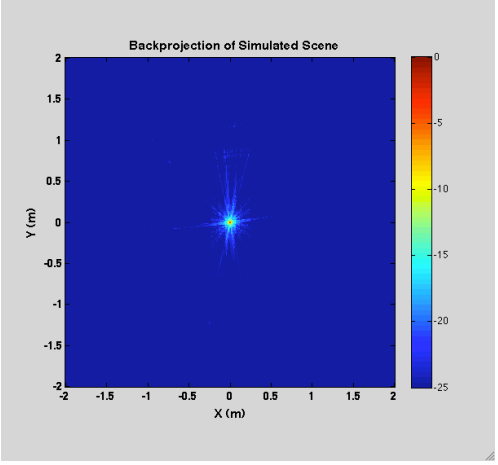

Figure 13: Back-Projection for geometry in Figure 12

When compared to Figure 12, it is seen that the Back-Projection is extremely accurate, with the projections adding constructively at the point location (0,0), which is the location of the object in the original image.

If Back-Projection is run for multiple objects in the scene, as shown in Figure 14 below, the Back-Projected image that is received is shown in Figure 15 below. Once again it is observed that the projections add constructively at the point of the objects, and the location of the objects in the reconstructed image is extremely accurate. However, there seems to be constructive addition of a lower magnitude at other points of the image as well, and this is due to the Point Spread Function. This means that because of an imperfect impulse response during the reconstruction and limited set of projections, instead of representing a point reflector exactly, the reconstructed image spreads out. However, as this is of a much smaller magnitude, the Point Spread Function does not have significant affect on the reconstructed image.

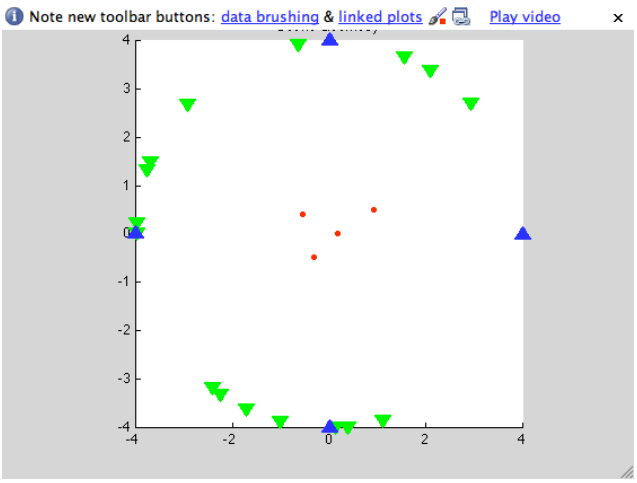

Figure 14: Simulated Test Bed for Bi-Static Imaging with four Reflectors

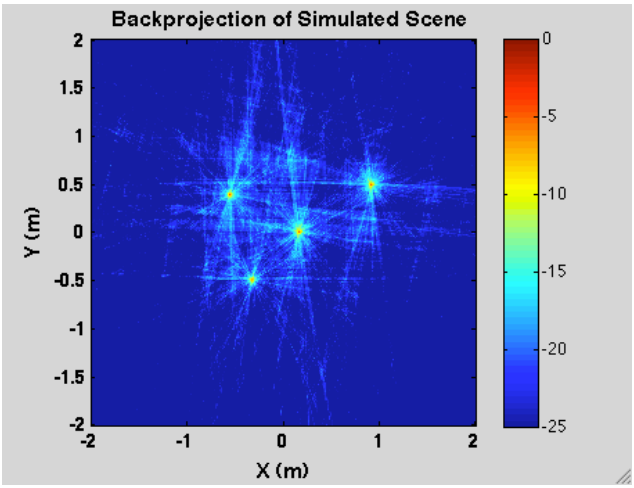

Figure 15: Back-Projection for geometry in Figure 14

6.2: Synthetic Aperture Radar/ Mono-Static Imaging

Back-Projection and FFBP were implemented on simulated data collected for the geometry found in SAR. This geometry is shown in Figure 16. The figure shown has 41 collocated transmitters and receivers in a linear array. On one side of this linear array, there are four point reflectors, and the objective is to reconstruct these using Back-Projection as well as FFBP for different Sub-Aperture values.

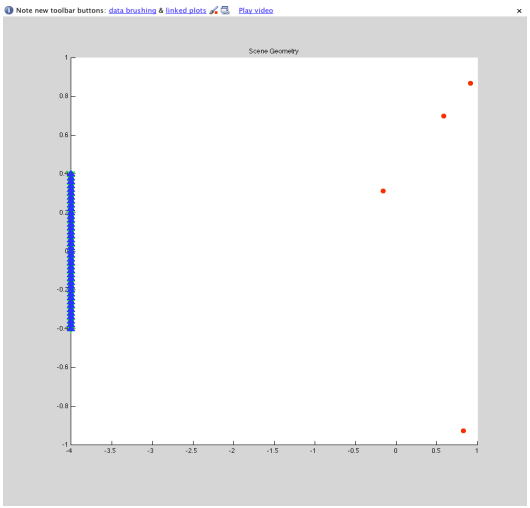

Figure 16: Simulated Test Bed for SAR Imaging with four Reflectors

Implementing Back-Projection and Fast-Factorized Back-Projection yielded some interesting results. When an image of N=1024 was reconstructed, it was seen that the computational time was most for Back-Projection, and it decreased for increasing amount of sub-apertures. These results can be seen in Table 1.

N=1024; nTx= nRx= 41; Point Reflectors= 4	Computational Time
Back- Projection	1049 Seconds
FFBP with 2 Sub-Apertures	972 Seconds
FFBP with 3 Sub-Apertures	411 Seconds
FFBP with 4 Sub-Apertures	377 Seconds
FFBP with 5 Sub-Apertures	236 Seconds
FFBP with 6 Sub-Apertures	147 Seconds
FFBP with 7 Sub-Apertures	129 Seconds
FFBP with 8 Sub-Apertures	113 Seconds

Table 1: Computational Times for Back-Projection and FFBP

However, when the images were compared qualitatively (Figures 17 through 20), it was observed that the quality of the reconstructed image decreased as the computational time decreased. Some of the things that were observed were that with increasing number of sub-apertures, the sharpness of the image went down. Moreover, with an increasing number of sub-apertures, instead of reconstructing a point reflector, the reconstructed images seemed to reconstruct objects that have a length. Therefore, for sub-apertures sizes larger then three, the reconstruction yielded a blurred or smeared point spread function.

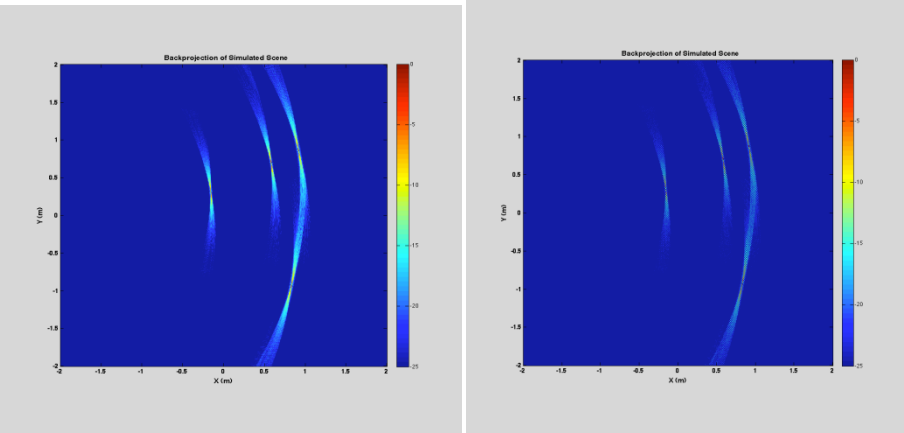

Figure 17: (a) Back-Projection and (b) FFBP for 2 Sub-Apertures

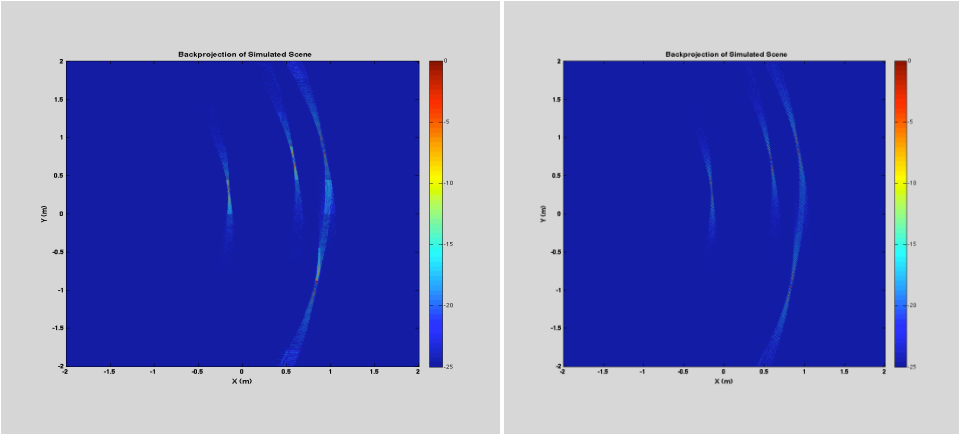

Figure 18: FFBP for (a) 3 and (b) 4 Sub-Apertures

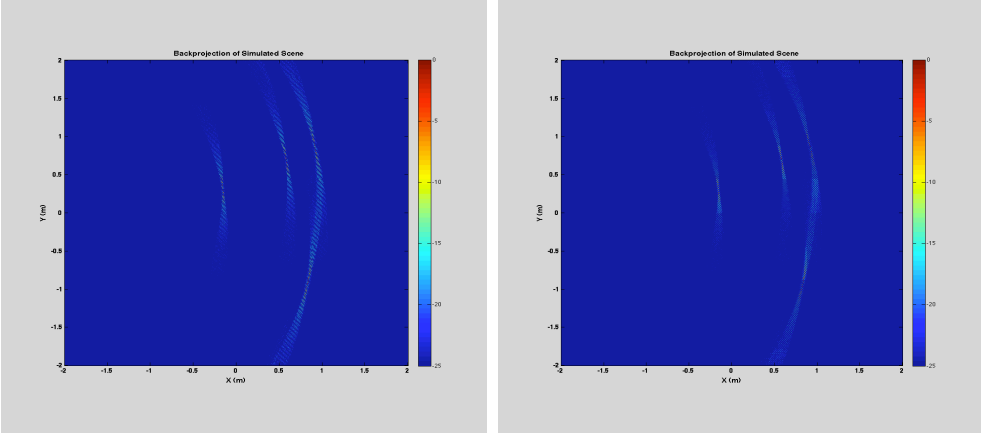

Figure 19: FFBP for (a) 5 and (b) 6 Sub-Apertures

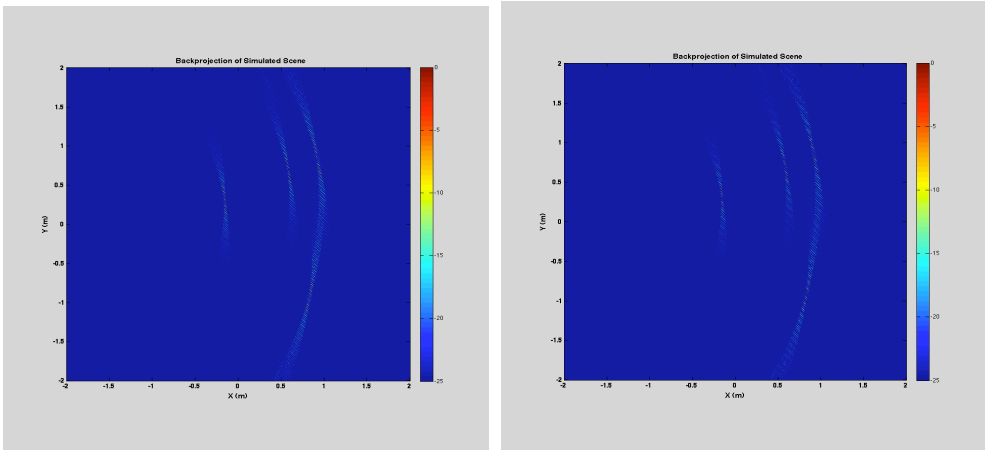

Figure 20: FFBP for (a) 7 and (b) 8 Sub-Apertures

Chapter 7: Conclusions and Future Work

The first part of this thesis was related to implementation of Back-Projection for a Bi-Static transmitter and receiver geometry. This was done for both simulated and experimental data. An acoustic imaging testbed was constructed for the experimental imaging. The testing of the Matlab codes on the data saw successful reconstruction. The reconstruction was successfully computed for a single point reflector, as well as multiple point reflectors.

The next part of the project was related to Synthetic Aperture Radar transmitter and receiver geometry. Using this geometry, data were simulated using Matlab, and Back-Projection and Fast Factorized Back-Projection were implemented on the simulated data. When the two imaging techniques were tested on the data, it was seen that Back-Projection was much slower, and the processing time for FFBP decreased with increasing number of Sub-Apertures. However, the increase in the number of Sub-Apertures resulted in the image quality of the reconstructed image going down. By doing a qualitative analysis of the reconstructed images, it was determined that if more then three sub-apertures were used, the image quality degrades sufficiently that the reconstructed image is no longer valid. However, even for three sub-apertures, the reduction in processing time is sufficiently large for FFBP to have a clear advantage over Back-Projection.

Chapter 8: References

[1] M. Soumekh, Synthetic Aperture Radar Signal Processing. John Wiley & Sons inc. USA, 1999, ISBN:0-471-29706-2.

[2] A. Olofsson, “Signalbehandling i flygburen ultrabredbandig l°agfrekvens-SAR,” Master’s thesis, Chalmers tekniska h¨ogskolan Institutionen f¨or radio- och rymdvetenskap, 2003.

[3] W. S. George, Introduction to Airborne Radar second edition. Scitech publishing.inc USA, 1998, ISBN:1891121014.

[4] L. M. H. Ulander, H. Hellsten, and G. Stenstr¨om,“Synthetic-Aperture Radar Processing Using Fast Factorized Back-Projection,” IEEE Transaction on aerospace and electronic systems, vol. 39, no. 3, Jul 2003.

[5] A. Hast and L. Johansson, *Fast factorized back-projection in an FPGA*, M.S. thesis, Halmstad University, Halmstad, Sweden, 2006, http://hdl.handle.net/2082/576.

Appendix A:

A1: Back-Projection Code:

```
%%Written by Taylor Williams 3/23/2011
%
% Code that takes correlated time-domain recordings and produces an image
% through backprojection
%
% Input Parameters:
%   cdata: cell structure where cdata{tx,rx} is the recording from the
%       specified Tx and Rx.  t=0 when the sound leaves the speaker.
%
%   Tx/Rx:  vector telling the positions of each transmitter and receiver
%       in the x-y plane using complex numbers (x+iy) in meters
%
%   L:  Width of the scene to image.  Produced image will span from
%       [-L/2, L/2] in both x and y directions
%
%   N:  Width in pixels of image to produce.  Function produces an NxN
%       image.
%
%
% Output Parameters:
%   pixelgrid:  NxN matrix where each entry is the location in the x-y
%       plane of the center of that pixel.
%
%   image:  NxN matrix with the result of the backprojection in the scene.
%       magnitudes normalized to be from 0 to 1.

function [pixelgrid image] = backproject(cdata, Tx, Rx, L, N)
    %Static Variables - speed of sound and sampling rate
    c = 343;
    fs = 96000;

    nTx = length(Tx); nRx = length(Rx);

    %Resample provided data vectors so that there is one point per pixel
    %  either interpolating or decimating by a rational factor

    R = (c/fs)/(L/N); %resampling factor (provided delta x / image delta x)

    %use interp1 to create a new interpolated data vector to fit N pixels
    %  This is alot faster to do ahead of time all in one swoop
    fprintf('Resampling Data...');
    for tx = 1:nTx
```

```
        len = length(cdata{tx,:});
        cdata{tx,:} = interp1(1:len,cdata{tx,:},linspace(1,len,fix(len*R)));

    end
    fprintf(' Complete.\n');

    %make sure N is odd (number of pixels in final image)
    %guarantees a (0,0) pixel
    if (mod(N,2)==0) N = N-1; end

    %create 2D NxN vector that contains the positions of the center of each
    %pixel (using complex numbers for x,y coordinates)
    pixpos = (linspace(-L/2,L/2,N)'*ones(1,N))'+linspace(-L/2*i,L/2*i,N)'*ones(1,N);

    %initialize image to zeros
    image = zeros(N,N);

    %Looping through each pixel in the final image
    fprintf('Constructing Image: ');
    tic;
    mark = 0;
    for x=1:N
       %display status
       t=toc;
       if (floor(t)>mark) fprintf('%d%%, ',ceil(x/N*100)); mark = mark + 1; end
       for y=1:N
          %examining the contribution from each geometry
          for tx=1:nTx
                %calculate the path distance from the chosen Tx/Rx to
                %the pixel being examined
                d = abs(Tx(tx)-pixpos(x,y))+abs(Rx(tx)-pixpos(x,y));
                n = fix(R*d/c*fs); %determine the index for the distance using the
resampling factor
                %add to the pixel the value from the range profile
                if (n <= length(cdata{tx,:})) image(x,y) = image(x,y) + abs(cdata{tx,:}(n));
end
          end
       end
    end
    fprintf(' Completed.\n');
    toc
    %normalize image
    image = image./max(max(image));
    pixelgrid = pixpos;

end
```

A2: FFBP for 2 Sub-Apertures:

```
%%Written by Danish Siddiqui by editing Back-Projection code written by
%%Taylor Williams
%
% Code that takes correlated time-domain recordings and produces an image
% through Fast-Factorized Back-Propjection for 2 Sub-Apertures
%
% Input Parameters:
%   cdata: cell structure where cdata{tx,rx} is the recording from the
%       specified Tx and Rx.  t=0 when the sound leaves the speaker.
%
%   Tx/Rx:  vector telling the positions of each transmitter and receiver
%       in the x-y plane using complex numbers (x+iy) in meters
%
%   L:  Width of the scene to image.  Produced image will span from
%       [-L/2, L/2] in both x and y directions
%
%   N:  Width in pixels of image to produce.  Function produces an NxN
%       image.
%
%
% Output Parameters:
%   pixelgrid:  NxN matrix where each entry is the location in the x-y
%       plane of the center of that pixel.
%
%   image:  NxN matrix with the result of the backprojection in the scene.
%       magnitudes normalized to be from 0 to 1.

function [pixelgrid image] = subap2(cdata, Tx, Rx, L, N)
    %Static Variables - speed of sound and sampling rate
    c = 343;
    fs = 96000;

    nTx = length(Tx);
    nRx = length(Rx);

    %Resample provided data vectors so that there is one point per pixel
    %  either interpolating or decimating by a rational factor

    R = (c/fs)/(L/N/2); %resampling factor (provided delta x / image delta x)

    %use interp1 to create a new interpolated data vector to fit N pixels
    %  This is alot faster to do ahead of time all in one swoop
    fprintf('Resampling Data...');
    for tx = 1:nTx
```

```
        len = length(cdata{tx,:});
        cdata{tx,:} = interp1(1:len,cdata{tx,:},linspace(1,len,fix(len*R)));

end
fprintf(' Complete.\n');

%make sure N is odd (number of pixels in final image)
%guarantees a (0,0) pixel
if (mod(N,2)==0) N = N-1; end

%create 2D NxN vector that contains the positions of the center of each
%pixel (using complex numbers for x,y coordinates)
pixpos = (linspace(-L/2,L/2,N)'*ones(1,N))'+linspace(-L/2*i,L/2*i,N)'*ones(1,N);
pixpos1=zeros(N/2,N/2);
pixpos2=zeros(N/2,N/2);

for y=0:N/2-1
   for z=0:N/2-1
      pixpos1(y+1,z+1)=pixpos(2*y+1,2*z+1);
      pixpos2(y+1,z+1)=pixpos(2*y+2,2*z+2);
   end
end

%initialize image to zeros
image = zeros(N,N);

 for j=1:length(cdata)
   lengths(j)=length(cdata{j,:});
 end
maxlength=max(lengths);

cdata1=zeros(length(cdata),maxlength);

for j=1:length(cdata)
   for k=1:length(cdata{j,:})
      cdata1(j,k)=abs(cdata{j,:}(k));
   end
end

l2=floor(length(cdata1)/2);
cdata11=zeros(length(cdata),l2);
cdata12=zeros(length(cdata),l2);


for j=0:(l2-1)
   for k=1:length(cdata)
```

```
        cdata11(k,j+1)=cdata1(k,2*j+1);
        cdata12(k,j+1)=cdata1(k,2*j+2);
    end
end

image1=zeros(N/2,N/2);
image2=zeros(N/2,N/2);


%Looping through each pixel in the final image
fprintf('Constructing Image: ');
tic;
mark = 0;
for x=0:N/2-1
    %display status
    t=toc;
    if (floor(t)>mark) fprintf('%d%%, ',ceil(x/N*100)); mark = mark + 1; end
    for y=0:N/2-1
        %examining the contribution from each geometry
        for tx=1:nTx
            %calculate the path distance from the chosen Tx/Rx to
            %the pixel being examined
            d = abs(Tx(tx)-pixpos1(x+1,y+1))+abs(Rx(tx)-pixpos1(x+1,y+1));
            n = fix(R*d/c*fs/2); %determine the index for the distance using the
resampling factor
            %add to the pixel the value from the range profile
            if (n <= length(cdata11(tx,:))) image1(x+1,y+1) = image1(x+1,y+1) +
abs(cdata11(tx,n)); end

        end
    end
end
toc

for x=0:N/2-1
    %display status
    t=toc;
    if (floor(t)>mark) fprintf('%d%%, ',ceil(x/N*100)); mark = mark + 1; end
    for y=0:N/2-1
        %examining the contribution from each geometry
        for tx=1:nTx
            %calculate the path distance from the chosen Tx/Rx to
            %the pixel being examined
            d = abs(Tx(tx)-pixpos2(x+1,y+1))+abs(Rx(tx)-pixpos2(x+1,y+1));
            n = fix(R*d/c*fs/2); %determine the index for the distance using the
resampling factor
```

```
                %add to the pixel the value from the range profile
                if (n <= length(cdata12(tx,:))) image2(x+1,y+1) = image2(x+1,y+1) +
abs(cdata12(tx,n)); end

            end
        end
    end


    fprintf(' Completed.\n');
    toc

    for v=0:N/2-1
        for s=0:N/2-1
            image(2*v+1,2*s+1)=image1(v+1,s+1);
            image(2*v+2,2*s+2)=image2(v+1,s+1);
        end
    end
    toc

    %normalize image
    image = image./max(max(image));

    pixelgrid = pixpos;

end
```

A3: FFBP for 3 Sub-Apertures:

```
%%Written by Danish Siddiqui by editing Back-Projection code written by
%%Taylor Williams
%
% Code that takes correlated time-domain recordings and produces an image
% through Fast-Factorized Back-Propjection for 2 Sub-Apertures
%
% Input Parameters:
%   cdata: cell structure where cdata{tx,rx} is the recording from the
%       specified Tx and Rx.  t=0 when the sound leaves the speaker.
%
%   Tx/Rx:  vector telling the positions of each transmitter and receiver
%       in the x-y plane using complex numbers (x+iy) in meters
%
%   L:  Width of the scene to image.  Produced image will span from
%       [-L/2, L/2] in both x and y directions
%
%   N:  Width in pixels of image to produce.  Function produces an NxN
%       image.
%
%
% Output Parameters:
%   pixelgrid:  NxN matrix where each entry is the location in the x-y
%       plane of the center of that pixel.
%
%   image:  NxN matrix with the result of the backprojection in the scene.
%       magnitudes normalized to be from 0 to 1.

function [pixelgrid image] = subap3(cdata, Tx, Rx, L, N)
    %Static Variables - speed of sound and sampling rate
    c = 343;
    fs = 96000;

    nTx = length(Tx);
    nRx = length(Rx);

    %Resample provided data vectors so that there is one point per pixel
    %  either interpolating or decimating by a rational factor

    R = (c/fs)/(L/N/3); %resampling factor (provided delta x / image delta x)

    %use interp1 to create a new interpolated data vector to fit N pixels
    %  This is alot faster to do ahead of time all in one swoop
    fprintf('Resampling Data...');
    for tx = 1:nTx
```

```
        len = length(cdata{tx,:});
        cdata{tx,:} = interp1(1:len,cdata{tx,:},linspace(1,len,fix(len*R)));

end
fprintf(' Complete.\n');

%make sure N is odd (number of pixels in final image)
%guarantees a (0,0) pixel
if (mod(N,2)==0) N = N-1; end

%create 2D NxN vector that contains the positions of the center of each
%pixel (using complex numbers for x,y coordinates)
pixpos = (linspace(-L/2,L/2,N)'*ones(1,N))'+linspace(-L/2*i,L/2*i,N)'*ones(1,N);
pixpos1=zeros(N/3,N/3);
pixpos2=zeros(N/3,N/3);
pixpos3=zeros(N/3,N/3);
for y=0:N/3-1
   for z=0:N/3-1
      pixpos1(y+1,z+1)=pixpos(3*y+1,3*z+1);
      pixpos2(y+1,z+1)=pixpos(3*y+2,3*z+2);
      pixpos3(y+1,z+1)=pixpos(3*y+3,3*z+3);
   end
end

%initialize image to zeros
image = zeros(N,N);

 for j=1:length(cdata)
    lengths(j)=length(cdata{j,:});
 end
maxlength=max(lengths);

cdata1=zeros(length(cdata),maxlength);

for j=1:length(cdata)
   for k=1:length(cdata{j,:})
      cdata1(j,k)=abs(cdata{j,:}(k));
   end
end

l3=floor(length(cdata1)/3);
cdata11=zeros(length(cdata),l3);
cdata12=zeros(length(cdata),l3);
cdata13=zeros(length(cdata),l3);

for j=0:(l3-1)
```

```
    for k=1:length(cdata)
        cdata11(k,j+1)=cdata1(k,3*j+1);
        cdata12(k,j+1)=cdata1(k,3*j+2);
        cdata13(k,j+1)=cdata1(k,3*j+3);
    end
end

image1=zeros(N/3,N/3);
image2=zeros(N/3,N/3);
image3=zeros(N/3,N/3);

%Looping through each pixel in the final image
fprintf('Constructing Image: ');
tic;
mark = 0;
for x=0:N/3-1
    %display status
    t=toc;
    if (floor(t)>mark) fprintf('%d%%, ',ceil(x/N*100)); mark = mark + 1; end
    for y=0:N/3-1
        %examining the contribution from each geometry
        for tx=1:nTx
            %calculate the path distance from the chosen Tx/Rx to
            %the pixel being examined
            d = abs(Tx(tx)-pixpos1(x+1,y+1))+abs(Rx(tx)-pixpos1(x+1,y+1));
            n = fix(R*d/c*fs/3); %determine the index for the distance using the
resampling factor
            %add to the pixel the value from the range profile
            if (n <= length(cdata11(tx,:))) image1(x+1,y+1) = image1(x+1,y+1) +
abs(cdata11(tx,n)); end

        end
    end
end
toc

for x=0:N/3-1
    %display status
    t=toc;
    if (floor(t)>mark) fprintf('%d%%, ',ceil(x/N*100)); mark = mark + 1; end
    for y=0:N/3-1
        %examining the contribution from each geometry
        for tx=1:nTx
            %calculate the path distance from the chosen Tx/Rx to
            %the pixel being examined
            d = abs(Tx(tx)-pixpos2(x+1,y+1))+abs(Rx(tx)-pixpos2(x+1,y+1));
```

```
            n = fix(R*d/c*fs/3); %determine the index for the distance using the
resampling factor
            %add to the pixel the value from the range profile
            if (n <= length(cdata12(tx,:))) image2(x+1,y+1) = image2(x+1,y+1) +
abs(cdata12(tx,n)); end

        end
    end
end
toc

for x=0:N/3-1
    %display status
    t=toc;
    if (floor(t)>mark) fprintf('%d%%, ',ceil(x/N*100)); mark = mark + 1; end
    for y=0:N/3-1
        %examining the contribution from each geometry
        for tx=1:nTx
            %calculate the path distance from the chosen Tx/Rx to
            %the pixel being examined
            d = abs(Tx(tx)-pixpos3(x+1,y+1))+abs(Rx(tx)-pixpos3(x+1,y+1));
            n = fix(R*d/c*fs/3); %determine the index for the distance using the
resampling factor
            %add to the pixel the value from the range profile
            if (n <= length(cdata13(tx,:))) image3(x+1,y+1) = image3(x+1,y+1) +
abs(cdata13(tx,n)); end

        end
    end
end
toc
fprintf(' Completed.\n');
toc

for v=0:N/3-1
    for s=0:N/3-1
        image(3*v+1,3*s+1)=image1(v+1,s+1);
        image(3*v+2,3*s+2)=image2(v+1,s+1);
        image(3*v+3,3*s+3)=image3(v+1,s+1);
    end
end
toc
%normalize image
image = image./max(max(image));
pixelgrid = pixpos;
end
```